Министерство образования и науки РФ

ПНИПУ

Кафедра вычислительной математики и механики

Курсовая работа

По дисциплине:

«Информационная безопасность»

На тему:

Криптоанализ шифра Виженера

Выполнил:

Паздерин Альберт

Группа:

ИСТ-12

Пермь, 2015 год

Оглавление

[Теория 3](#_Toc419688889)

[Принцип шифра Виженера 3](#_Toc419688890)

[Криптоанализ шифра Виженера 5](#_Toc419688891)

[Метод Касиски 5](#_Toc419688892)

[Тест Фридмана 6](#_Toc419688893)

[Практика 10](#_Toc419688894)

[Особенности программы 10](#_Toc419688895)

[Криптоанализ обычного шифра Виженера 10](#_Toc419688896)

[Криптоанализ шифра Виженера с перемешанным алфавитом 12](#_Toc419688897)

[Заключение 15](#_Toc419688898)

[Листинги 15](#_Toc419688899)

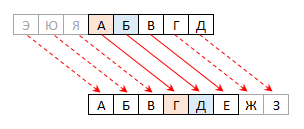
# **Теория**

Шифр Виженера (фр. Chiffre de Vigenère) — метод полиалфавитного шифрования буквенного текста с использованием ключевого слова.

Этот метод является простой формой многоалфавитной замены. Шифр Виженера изобретался многократно. Впервые этот метод описал Джован Баттиста Беллазо (итал. Giovan Battista Bellaso) в книге La cifra del. Sig. Giovan Battista Bellasо в 1553 году, однако в XIX веке получил имя Блеза Виженера, французского дипломата. Метод прост для понимания и реализации, он является недоступным для простых методов криптоанализа.

## **Принцип шифра Виженера**

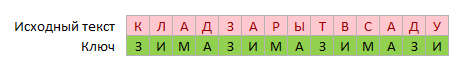
Одним из самых древних и простых в реализации является шифр Цезаря, который использовал его для тайной переписки. Суть его в том, что каждая буква исходного шифруемого сообщения сдвигается в алфавите на заданное количество символов. Так, например, если сдвиг равен 3, то буква А превратится в Г, буква Б - в Д и так далее:



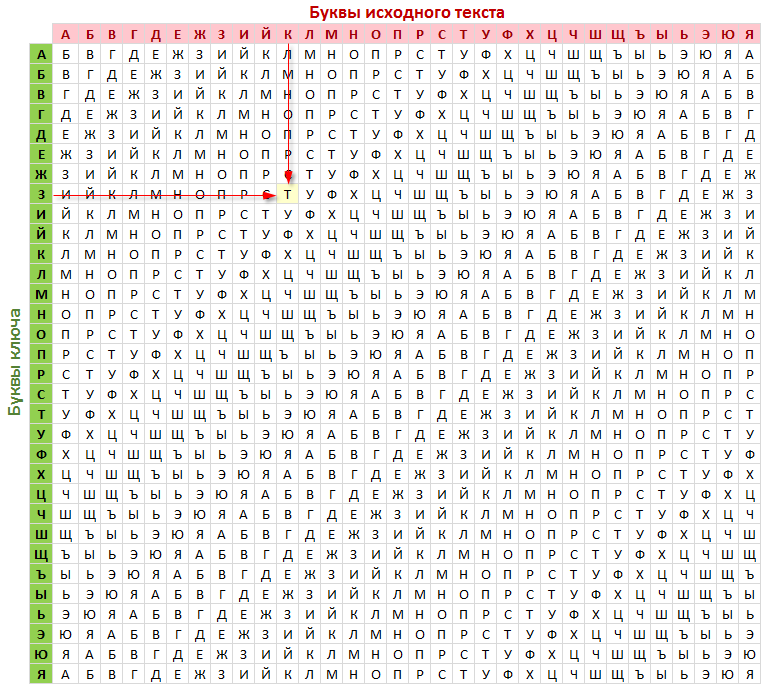
Символы в конце алфавита (Э, Ю, Я), соответственно, будут превращаться его начало (А, Б, В).

Реализовать такой шифр просто, но стойкость его невелика - найти нужное число сдвига и дешифровать сообщение можно даже прямым перебором за 20-30 итераций, что займет даже у человека не больше часа, а у современного компьютера доли секунды. Поэтому еще в 15 веке был впервые придуман, а потом в 16 веке французским дипломатом Блезом Виженером официально представлен более совершенный метод на основе шифра Цезаря, получивший впоследствии название "шифр Виженера". Его принцип в том, что каждая буква в исходном шифруемом тексте сдвигается по алфавиту не на фиксированное, а переменное количество символов. Величина сдвига каждой буквы задается ключом (паролем) - секретным словом или фразой, которая используется для шифрования и расшифровки.

Допустим, мы хотим зашифровать фразу "*КЛАД ЗАРЫТ В САДУ*" используя слово *ЗИМА* в качестве ключа. Запишем это слово подряд несколько раз под исходной фразой:



Для удобства шифрования используем так называемый "квадрат Виженера" - таблицу, где в каждой строке алфавит сдвигается на одну позицию вправо:

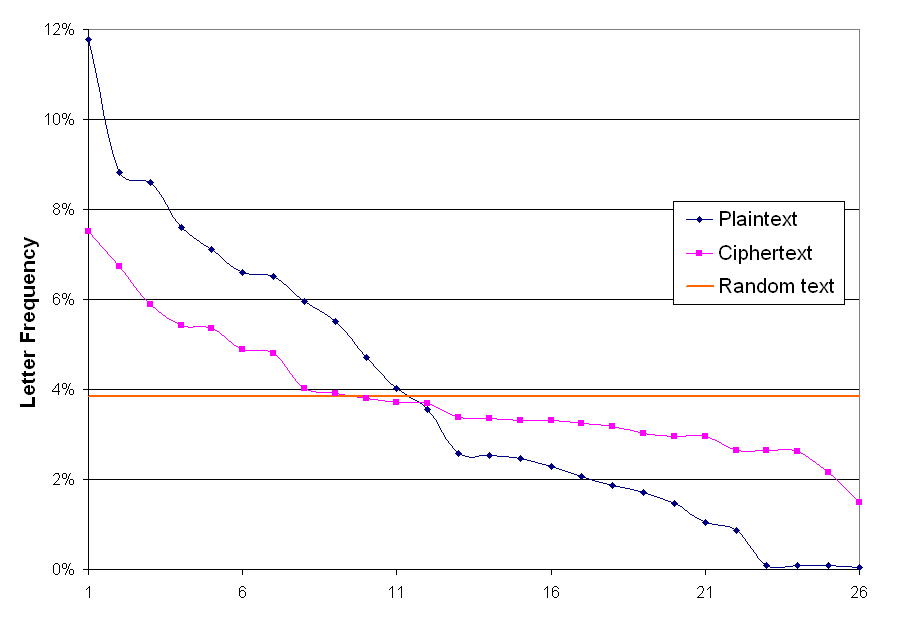


Если взять строку с первой буквой ключа (*З*) и столбец с первой буквой исходного текста (*К*), то на их пересечении увидим букву "*Т*" - это и будет первая буква нашего зашифрованного сообщения. Затем процедура повторяется для всех остальных пар букв ключа и исходного сообщения по очереди и в результате мы получаем зашифрованный вариант нашей исходной фразы:



Заметим, что одна и та же буква (например *А*) в исходном сообщений превратилась в разные буквы на выходе (*Н*, *Й* и *Б*), т.к. сдвиг при шифровании для них был разный. Именно поэтому вскрыть шифр Виженера простыми способами невозможно - вплоть до 19 века он считался невзламываемым и успешно использовался военными, дипломатами и шпионами многих стран, частности - конфедератами во время Гражданской войны в США.

## **Криптоанализ шифра Виженера**



Шифр Виженера «размывает» характеристики частот появления символов в тексте, но некоторые особенности появления символов в тексте остаются. Главный недостаток шифра Виженера состоит в том, что его ключ повторяется. Поэтому простой криптоанализ шифра может быть построен в два этапа:

1. **Поиск длины ключа.** Можно анализировать распределение частот в зашифрованном тексте с различным прореживанием. То есть брать текст, включающий каждую 2-ю букву зашифрованного текста, потом каждую 3-ю и т. д. Как только распределение частот букв будет сильно отличаться от равномерного (например, по энтропии), то можно говорить о найденной длине ключа.
2. **Криптоанализ.** Совокупность l шифров Цезаря (где l — найденная длина ключа), которые по отдельности легко взламываются.

Тесты Фридмана и Касиски могут помочь определить длину ключа.

### **Метод Касиски**

В 1863 году Фридрих Касиски был первым, кто опубликовал успешный алгоритм атаки на шифр Виженера, хотя Чарльз Беббидж разработал этот алгоритм уже в 1854 году. В то время когда Беббидж занимался взломом шифра Виженера, Джон Холл Брок Твейтс представил новый шифр в «Journal of the Society of the Arts»; когда Беббидж показал, что шифр Твейтса является лишь частным случаем шифра Виженера, Твейтс предложил ему его взломать. Беббидж расшифровал текст, который оказался поэмой «The Vision of Sin» Альфреда Теннисона, зашифрованной ключевым словом Emily — именем жены поэта.

Тест Касиски опирается на то, что некоторые слова, такие как «*the*» могут быть зашифрованы одинаковыми символами, что приводит к повторению групп символов в зашифрованном тексте. Например: сообщение, зашифрованное ключом *ABCDEF* , не всегда одинаково зашифрует слово «*crypto*»:

|  |  |
| --- | --- |
| Ключ | ABCDEF AB CDEFA BCD EFABCDEFABCD |
| Исходный текст | CRYPTO IS SHORT FOR CRYPTOGRAPHY |
| Шифрованный текст | CSASXT IT UKSWT GQU GWYQVRKWAQJB |

Зашифрованный текст в данном случае не будет повторять последовательности символов, которые соответствуют повторным последовательностям исходного текста. В данном шифрованном тексте есть несколько повторяющихся сегментов, которые позволяют криптоаналитику найти длину ключа:

|  |  |
| --- | --- |
| Ключ | ABCDAB CD ABCDA BCD ABCDABCDABCD |
| Исходный текст | CRYPTO IS SHORT FOR CRYPTOGRAPHY |
| Шифрованный текст | CSASTP KV SIQUT GQU CSASTPIUAQJB |

Более длинные сообщения делают тест более точным, так как они включают в себя больше повторяющихся сегментов зашифрованного текста. В данном шифрованном тексте есть несколько повторяющихся сегментов, которые позволяют криптоаналитику найти длину ключа:

|  |  |
| --- | --- |
| Шифрованный текст | DYDUXRMHTVDVNQDQNWDYDUXRMHARTJGWNQD |

Расстояние между повторяющимися DYDUXRMH равно 18, это позволяет сделать вывод, что длина ключа равна одному из значений: 18, 9, 6, 3 или 2. Расстояние между повторяющимися NQD равно 20. Из этого следует, что длина ключа равна 20, 10, 5, 4 или 2. Сравнивая возможные длины ключей, можно сделать вывод, что длина ключа (почти наверняка) равна 2.

### **Тест Фридмана**

Тест Фридмана (иногда называемый каппа-тестом) был изобретен Вильямом Фридманом в 1920 году. Фридман использовал индекс совпадения, который измеряет частоты повторения символов, чтобы взломать шифр.

**Индекс совпадений**

Рассмотрим текст, написанный на некотором языке. Алфавит данного языка будем полагать состоящим из m букв. Рассмотрим достаточно длинную строку \vec x из n букв. Если f_i задаёт количество i-той буквы алфавита в строке \vec x, то можно определить индекс совпадений как вероятность совпадения двух произвольных букв в строке:

I \left( \vec x \right) = \sum\limits_i {f_i \frac{{f_i  - 1}}{{n\left( {n - 1} \right)}}} 

откуда при достаточно больших n и определении p_i как p_i = f_i / n получаем приближённую формулу:

I \left( \vec x \right) = \sum\limits_i p_i^2 

Теперь, если предположить, что наш текст написан на естественном языке (например, на русском или английском), является достаточно длинным и его характеристики соответствуют известным нам характеристикам данного языка, то для него можно получить ожидаемый индекс совпадений (используя известные частоты появления отдельных букв):

| Язык | Индекс совпадений |
| --- | --- |
| русский | 0,0553 |
| английский | 0,0644 0.0667 |
| итальянский | 0.0738 |
| испанский | 0.0775 |
| немецкий | 0.0762 |
| французский | 0.0778 |

Для работы метода индекса совпадений важно, что данная характеристика не меняется, если текст на языке зашифрован с использованием моноалфавитного шифра, такого, как шифр Цезаря.

Для случайной строки символов алфавита это значение примерно равно

I \left( \vec x \right) \approx {1 \over m}

то есть 0,03125 для русского языка (без буквы «ё») и 0,03856 для английского языка.

Важным свойством является то, что индекс совпадения для строки не меняется при шифровании этой строки моноалфавитным шифром.

**Взаимный индекс совпадений**

Теперь рассмотрим две строки \vec x и \vec y с длинами n и n' соответственно, составленные из букв некоторого алфавита. Алфавит также имеет m букв. **Взаимным индексом совпадений** этих строк будем называть вероятность того, что буква, случайно выбранная из первой строки, совпадёт со случайно выбранной буквой второй строки.

Пусть f_i, g_i — количество i-ой буквы алфавита в первой и второй строке соответственно. Тогда взаимный индекс совпадений будет равен:

MI\left( {\vec x,\vec y} \right) = \frac{{\sum\limits_i {f_i g_i } }}{{nn'}}

Достаточно легко показать следующие свойства:

1. Если строки относятся к открытым текстам, написанным на одном и том же языке, то их взаимный индекс будет примерно равен табличному индексу для этого языка
2. Если строки совпадают, то их взаимный индекс совпадений примерно равен индексу совпадений (с точностью до 1/n)
3. Если хотя бы одна из строк является строкой из случайных символов, то взаимный индекс совпадений равен индексу совпадений для случайной строки, то есть 1 / m
4. При шифровании обоих сравниваемых строк произвольным шифром подстановки (но одинаковым для обоих строк) взаимный индекс не изменяется
5. Если из первой строки \vec x выбрать достаточно большую группу символов \vec x', с такими же частотами символов p^x_i, как и в исходной строке, то её взаимный индекс со второй строкой MI \left( \vec x', \vec y \right) будет примерно равен взаимному индексу оригинальной строки и второй строки MI \left( \vec x, \vec y \right)

**Взаимный индекс совпадений для зашифрованных строк**

Пусть \vec y_i, \vec y_j — строки, зашифрованные шифром Виженера с ключевым словом \vec k= \overrightarrow {k_1 ,k_2 ,...,k_k } . Можно показать, что для этих строк величина индекса совпадений примерно равна:

MI\left( {\vec y_i ,\vec y_j } \right) \approx \sum\limits_i^m {p_{i - k_i } p_{i - k_j } }  = \sum\limits_i^m {p_i p_{h + k_i  - k_j } }

Правая часть выражения зависит только от величины k_i - k_j, которая называется относительным сдвигом строк \vec y_i и \vec y_j. С учётом очевидного равенства

\sum\limits_i^m {p_i p_{i - \alpha } }  = \sum\limits_i^m {p_i p_{i + \alpha } } 

мы можем построить таблицу относительных сдвигов для любого \alpha от 1 до m/2, используя их для нахождения значений и в случае m / 2 < \alpha < m .

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Для русского языка:   |  |  | | --- | --- | | Сдвиг | Взаимный индекс | | 0 | 0,0553 | | 1 | 0,0366 | | 2 | 0,0345 | | 3 | 0,0400 | | 4 | 0,0340 | | 5 | 0,0360 | | 6 | 0,0326 | | 7 | 0,0241 | | 8 | 0,0287 | | 9 | 0,0317 | | 10 | 0,0265 | | 11 | 0,0251 | | 12 | 0,0244 | | 13 | 0,0291 | | 14 | 0,0322 | | 15 | 0,0244 | | 16 | 0,0249 | | Для английского языка:   |  |  | | --- | --- | | Сдвиг | Взаимный индекс | | 0 | 0,0644 | | 1 | 0,0394 | | 2 | 0,0319 | | 3 | 0,0345 | | 4 | 0,0436 | | 5 | 0,0332 | | 6 | 0,0363 | | 7 | 0,0389 | | 8 | 0,0338 | | 9 | 0,0342 | | 10 | 0,0378 | | 11 | 0,0440 | | 12 | 0,0387 | | 13 | 0,0428 | |

Стоит отметить, что хотя значения для ненулевых сдвигов что для русского, что для английского языка различаются между собой не очень сильно, они достаточно хорошо отличаются от значения индекса при нулевом сдвиге.

**Метод индекса совпадений**

Возьмём строку \vec y и последовательно циклически сдвигая строку на m/2 позиций построим новые m/2-1 строк. Найдём индексы взаимных совпадений между исходной строкой и каждой полученной. Максимальный индекс совпадений между двумя строками будет соответствовать случаю использования одинаковых шифров сдвига для соответствующих букв, то есть в случае когда размер сдвига кратен размеру ключевого слова шифра Виженера. Если строка достаточно длинная (то есть статистика достаточно точная), то можно сравнивать промежуточные значения со значениями из таблиц выше для определения размера сдвига и последующей проверки.

# **Практика**

## **Особенности программы**

* ЯП Python 3.
* Состоит из 3 модулей
  + encrypter.py – шифрование текста
  + analyzer.py – анализ текста
  + statistics.py – все, что связано со статистическими данными, а также со сбором статистики
* Использует английский алфавит: *abcdefghijklmnopqrstuvwxyz*
* Статистика частот символов взята из книги Роберта Леванда «Cryptological Mathematics»[[1]](#footnote-1). Также присутствует статистика, полученная из трилогии «Властелин Колец» (общий объем файла составляет 2 882 964 байт)[[2]](#footnote-2).
* Используется эталонный индекс совпадения 0.0667, предложенный Фридманом
* Программа шифрует и анализирует текстовые файлы обычным методом Виженера и методом Виженера с перемешанным алфавитом. В первом случае результатом анализа является ключ, а во втором – *n* таблиц замен символов, где *n* – длина ключа
* Для определения длины ключа используется тест Фридмана

## **Криптоанализ обычного шифра Виженера**

1. **Шифрование**

При помощи программы зашифруем[[3]](#footnote-3) текстовый файл[[4]](#footnote-4) размером 978 491 байт (955 КБ) с ключом «***qwertyu***». Файл является рассказом Стивена Кинга «Бесплодные земли». Заглавные символы в файле заменены на их строчные варианты. Приведем отрывки исходного и зашифрованного текстов:

Исходный текст:

|  |
| --- |
| egan to open before his eyes. it disclosed a dark scarlet furnace, petal upon secret petal, each blazing with its own secret fury. jake had never seen anything so beautiful, so intensely and utterly alive. now, as he stretched one grime-streaked hand out toward this wonder, the voices began to sing his own name… and a dreadful, deadly fear began to steal in toward the center of his heart. it was as cold as black ice and as heavy as stone |

Зашифрованный текст:

|  |
| --- |
| mqee fg ipng tmrqap nie nvuc. wy vmyutdbur w othk ecrcnqx zxcluos, eubql xekh eapcub haytn, sepm tfwfjgs uolk qby qvg cmunna pwfg. pugs ktv dsjnc cmsr tlwxpwen cc dntebirkg, cc weauhystt yhn kyaudfy riqzm. efc, wm mu ydnnawnmt dgu kfwfp-cxfarjur peeq gwd yfcuzt ymqc aqequd, lkp zcocnb tmqee fg ewen nie dxl dwsn… udt r vdmwbmzn, tarqnw rarc tmqee fg elntn ij yfcuzt ymu osryph cr klg jseaa. id vtg qe pfnr wm snuom jru qjb tg jsewv uy myflq |

Вид квадрата Виженера:

a b c d e f g h i j k l m n o p q r s t u v w x y z

b c d e f g h i j k l m n o p q r s t u v w x y z a

c d e f g h i j k l m n o p q r s t u v w x y z a b

d e f g h i j k l m n o p q r s t u v w x y z a b c

e f g h i j k l m n o p q r s t u v w x y z a b c d

f g h i j k l m n o p q r s t u v w x y z a b c d e

g h i j k l m n o p q r s t u v w x y z a b c d e f

h i j k l m n o p q r s t u v w x y z a b c d e f g

i j k l m n o p q r s t u v w x y z a b c d e f g h

j k l m n o p q r s t u v w x y z a b c d e f g h i

k l m n o p q r s t u v w x y z a b c d e f g h i j

l m n o p q r s t u v w x y z a b c d e f g h i j k

m n o p q r s t u v w x y z a b c d e f g h i j k l

n o p q r s t u v w x y z a b c d e f g h i j k l m

o p q r s t u v w x y z a b c d e f g h i j k l m n

p q r s t u v w x y z a b c d e f g h i j k l m n o

q r s t u v w x y z a b c d e f g h i j k l m n o p

r s t u v w x y z a b c d e f g h i j k l m n o p q

s t u v w x y z a b c d e f g h i j k l m n o p q r

t u v w x y z a b c d e f g h i j k l m n o p q r s

u v w x y z a b c d e f g h i j k l m n o p q r s t

v w x y z a b c d e f g h i j k l m n o p q r s t u

w x y z a b c d e f g h i j k l m n o p q r s t u v

x y z a b c d e f g h i j k l m n o p q r s t u v w

y z a b c d e f g h i j k l m n o p q r s t u v w x

z a b c d e f g h i j k l m n o p q r s t u v w x y

1. **Поиск длины ключа**

Первым делом программа подсчитывает[[5]](#footnote-5) число символов алфавита в зашифрованном тексте. Затем по частоте символов определяется[[6]](#footnote-6) их доли, а по долям – кандидаты на длину ключа.

Доли символов алфавита:

1: 0.02262911104583483,

2: 0.022382149112786534,

3: 0.03803540150353036,

4: 0.04808142128224376,

5: 0.04498732552193827,

6: 0.039926237801494825,

7: 0.06368571646159035,

8: 0.0435490714448905,

9: 0.044342178269525766,

10: 0.04412023891119162,

11: 0.04453256742495947,

12: 0.04372858122001371

Далее пользователю предлагается выбрать[[7]](#footnote-7) длину ключа. Из таблицы видно, что наиболее подходящая длина ключа – 7.

1. **Нахождение ключа**

Зашифрованный текст разбивается на 7 групп, каждая из которых по сути

является шифротекстом метода Цезаря. Дальнейшая задача – нахождение ключа с помощью частотного анализа. Подсчитываются[[8]](#footnote-8) частоты появления символов алфавита в каждой группе, которые сравниваются со статистическими частотами. Для символов каждой группы ставятся в соответствие самые близкие по частотам символы и определяется наиболее частый сдвиг. Найденные сдвиги для каждой группы являются символами ключа. Полученный[[9]](#footnote-9) ключ может быть использован для расшифровки текста.

## **Криптоанализ шифра Виженера с перемешанным алфавитом**

Особенностью анализа такого шифротекста является то, что мы не можем однозначно определить ключ ввиду незнания последовательности символов исходного алфавита. Процесс криптоанализа будет повторять процесс анализа шифротекста без перестановки алфавита, за исключением последнего этапа, где нам придется искать таблицы перестановки и применять их к шифротексту.

1. **Шифрование**

Зашифруем файл с ключом «***asdf***».

Исходный текст:

|  |
| --- |
| an to open before his eyes. it disclosed a dark scarlet furnace, petal upon secret petal, each blazing with its own secret fury. jake had never seen anything so beautiful, so intensely and utterly alive. now, as he stretched one grime-streaked hand out toward this wonder, the voices began to sing his own name… and a dreadful, deadly fear began to steal in toward the center of his heart. it was as cold as black ice and as heavy as stone |

Зашифрованный текст:

|  |
| --- |
| za qe froa nolfpo nsv uxoz. wo uwzywezuu n uzbr vdvpbxm xpkbzdu, uuqzf dmth vxypoo roovw, umhj nbndsdq lwon woq emh vxypoo ztbc. hvjo nmn hvsxk vxub nhxujpbr qh rumtopztf, vt sdoubvxgx nhu pmqobgx ngssx. dtj, nq cx iubuqhjuu thv qkspx-iubumqxz cnhu teq oflzbz ujpi mfbnxk, onv kfshxq lxlmd mh zpbr nsv fld hmpx… zaz z zponzztf, nxvubi zonk lxlmd mh zmvzf sd mhinku onv dubuxk el owz oonkq. pq mvi nq htgu nq lfvgq pgo vbn vi jumsi mv qqeau |

Вид квадрата Виженера:

z g c f i t u w x h r a v k n b j e m p o y l d q s

g c f i t u w x h r a v k n b j e m p o y l d q s z

c f i t u w x h r a v k n b j e m p o y l d q s z g

f i t u w x h r a v k n b j e m p o y l d q s z g c

i t u w x h r a v k n b j e m p o y l d q s z g c f

t u w x h r a v k n b j e m p o y l d q s z g c f i

u w x h r a v k n b j e m p o y l d q s z g c f i t

w x h r a v k n b j e m p o y l d q s z g c f i t u

x h r a v k n b j e m p o y l d q s z g c f i t u w

h r a v k n b j e m p o y l d q s z g c f i t u w x

r a v k n b j e m p o y l d q s z g c f i t u w x h

a v k n b j e m p o y l d q s z g c f i t u w x h r

v k n b j e m p o y l d q s z g c f i t u w x h r a

k n b j e m p o y l d q s z g c f i t u w x h r a v

n b j e m p o y l d q s z g c f i t u w x h r a v k

b j e m p o y l d q s z g c f i t u w x h r a v k n

j e m p o y l d q s z g c f i t u w x h r a v k n b

e m p o y l d q s z g c f i t u w x h r a v k n b j

m p o y l d q s z g c f i t u w x h r a v k n b j e

p o y l d q s z g c f i t u w x h r a v k n b j e m

o y l d q s z g c f i t u w x h r a v k n b j e m p

y l d q s z g c f i t u w x h r a v k n b j e m p o

l d q s z g c f i t u w x h r a v k n b j e m p o y

d q s z g c f i t u w x h r a v k n b j e m p o y l

q s z g c f i t u w x h r a v k n b j e m p o y l d

s z g c f i t u w x h r a v k n b j e m p o y l d q

1. **Поиск длины ключа**

Доли символов алфавита:

1: 0.02378749301998351,

2: 0.023610094766953124,

3: 0.036543032179639924,

4: 0.05780864776165738,

5: 0.04392320268355176,

6: 0.03608340943315211,

7: 0.04648842174015591,

8: 0.0528072265597439,

9: 0.04282051126982879,

10: 0.040391364736855895,

11: 0.04290921039634398,

12: 0.05282738545213372

Ключи, близкие к эталонному, находятся на позициях 4, 8 и 12. Очевидно, что длиной ключа является 4.

1. **Нахождение таблиц замен**

Аналогично предыдущему испытанию, шифротекст разбивается на 4 группы и сравнением со статистическими частотами находятся[[10]](#footnote-10) 4 таблицы замен. Далее таблицы применяются[[11]](#footnote-11) к зашифрованному файлу и формируется новый файл. Для сравнения приведем отрывки из исходного файла и расшифрованного.

Исходный текст:

|  |
| --- |
| it began to open before his eyes. It disclosed a dark scarlet furnace, petal upon secret petal, each blazing with its own secret fury. Jake had never seen anything so beautiful, so intensely and utterly alive. Now, as he stretched one grime-streaked hand out toward this wonder, the voices began to sing his own name… and a dreadful, deadly fear began to steal in toward the center of his heart. It was as cold as black ice and as heavy as stone |

Расшифрованный текст:

|  |
| --- |
| nt bewan to oben beyore iss epeh. Nt rnhgdoher a rarb sfardet pcrnafe, betad cbon segret betad, eafn bdaqsnw unti nth omn segret pcrp. Jabe iar neker seen anptnnnw ho beactnpcd, so sntensedp anr ctterdp adske. Nom, ah ne htretfner one wrsfe-htreaber nanr oct touarr tnnh monrer, tie kosfeh bewan to hnnw iss oun nafe… anr a rrearpcd, reardp pear bewan to htead sn tomarr tie fenter oy nnh neart. Nt mah ah fodr ah bdafb nfe anr ah neakp as htone. |

При помощи одной из функции[[12]](#footnote-12) было определено, что исходный текст соответствует расшифрованному на 40.6% (без учета символов, не входящих в алфавит).

# **Заключение**

Существуют другие, возможно, более эффективные методы криптоанализа шифра Виженера. Например, генетический алгоритм посредством частотного анализа определяет примерную перестановку исходного алфавита, по которой затем строится квадрат Виженера и ищется ключ.

Для подтверждения найденой тестом Фридмана длины ключа может быть полезно дополнительно воспользаваться методом Касиски.

# **Листинги**

**encrypter.py**

**class** VigenereSquare(): *#Класс представляет квадрат виженера* **def** \_\_init\_\_(self, alphabet, is\_shuffle=**False**):  
 self.square = []  
 **from** copy **import** copy  
 current\_alphabet = copy(alphabet)  
 **if** is\_shuffle:  
 **from** random **import** shuffle  
 shuffle(current\_alphabet)  
 self.square.append(current\_alphabet)  
 **for** i **in** range(1, len(current\_alphabet)):  
 current\_alphabet = copy(current\_alphabet)  
 self.square.append(self.shift(current\_alphabet))  
  
  
 @staticmethod  
 **def** shift(l):  
 t = l[0]  
 **for** i **in** range(len(l) - 1):  
 l[i] = l[i + 1]  
 l[len(l) - 1] = t  
 **return** l  
  
 **def** \_\_str\_\_(self):  
 result = **""  
 for** alphabet **in** self.square:  
 **for** c **in** alphabet:  
 result += c + **' '** result += **'\n'  
 return** result  
  
 **def** \_\_getitem\_\_(self, c):  
 i, j = c  
 index = self.square[0].index(j)  
 index2 = self.square[index].index(i)  
 **return** self.square[0][index2]  
  
  
**class** VigenereEncrypter(): *#Шифрование текста* **def** \_\_init\_\_(self):  
 self.alphabet = statistics.english\_alphabet  
 self.square = VigenereSquare(self.alphabet, **False**)  
 self.key\_index = 0  
  
 **def** encrypt(self, text, key):  
 encrypted\_text = **""  
 for** i **in** range(len(text)):  
 **if** text[i] **in** self.alphabet:  
 encrypted\_text += self.square[key[self.key\_index % len(key)], text[i]]  
 **else**:  
 encrypted\_text += text[i]  
 self.key\_index += 1  
 **return** encrypted\_text

**analyzer.py**

**class** VigenereAnalyzer(): *#Анализ шифротекста* **def** \_\_init\_\_(self):  
 self.stats\_frequency = statistics.frequency\_english  
 self.alphabet = statistics.english\_alphabet  
 self.alphabet\_indexes = statistics.english\_alphabet\_indexes  
 self.alphabet\_values = statistics.english\_alphabet\_values  
 self.index = 0.0667  
 self.matches\_table\_length = int(len(self.alphabet) / 2)  
 self.matches\_table = {offset: 0  
 **for** offset  
 **in** range(1, self.matches\_table\_length)} *#range(1, int(len(self.alphabet) / 2))* self.key\_length = **None** self.replacement\_tables = []  
 self.frequency\_tables = []  
 self.key\_index = 0

@staticmethod  
 **def** shift(text, l):  
 result = **""  
 for** i **in** range(len(text) - l):  
 result += text[i + l]  
 **for** i **in** range(l):  
 result += text[i]  
 **return** result  
  
 **def** add\_matches\_info(self, text):  
 *#if len(text) < int(len(self.frequency) / 2):  
 #raise Exception("String too short")* shift = self.matches\_table\_length **if** self.matches\_table\_length <= len(text) **else** len(text)  
 **for** i **in** range(1, shift):  
 shifted = self.shift(text, i)  
 **for** j **in** range(len(text)):  
 **if** text[j] == shifted[j] **and** text[i] **in** self.alphabet:  
 self.matches\_table[i] += 1  
  
 **def** add\_frequency\_info(self, text):  
 **for** i **in** range(len(text)):  
 **if** text[i] **in** self.alphabet:  
 self.frequency\_tables[self.key\_index % self.key\_length][text[i]] += 1  
 self.key\_index += 1  
  
 **def** decode(self, text):  
 result = **""  
 for** i **in** range(len(text)):  
 **if** text[i] **in** self.alphabet:  
 result += self.replacement\_tables[self.key\_index % self.key\_length][text[i]]  
 **else**:  
 result += text[i]  
 self.key\_index += 1  
 **return** result  
  
 **def** count\_matches(self):  
 total\_matches = 0  
 **for** offset **in** self.matches\_table.keys():  
 total\_matches += self.matches\_table[offset]  
 **for** offset **in** self.matches\_table.keys():  
 self.matches\_table[offset] /= total\_matches  
 self.matches\_table[offset] /= 2  
  
 **def** count\_frequencies(self):  
 **for** table **in** self.frequency\_tables:  
 total\_chars = 0  
 **for** c **in** table.keys():  
 total\_chars += table[c]  
 **for** c **in** table.keys():  
 table[c] = 100 \* table[c] / total\_chars  
  
  
 **def** set\_key\_length(self, key\_length):  
 self.key\_length = key\_length  
 **for** i **in** range(self.key\_length):  
 self.frequency\_tables.append({c: 0 **for** c **in** self.alphabet})  
  
 **def** make\_replacement\_tables(self):  
 **for** i **in** range(self.key\_length):  
 current\_table = {c: **None for** c **in** self.alphabet}  
 **for** c\_table **in** current\_table.keys():  
 min\_d = 100.0  
 min\_c = **None  
 for** c\_freq **in** self.frequency\_tables[i]:  
 current\_d = abs(self.frequency\_tables[i][c\_table] - self.stats\_frequency[c\_freq])  
 **if** current\_d < min\_d:  
 min\_d = current\_d  
 min\_c = c\_freq  
 current\_table[c\_table] = min\_c  
 self.replacement\_tables.append(current\_table)  
  
 **def** find\_key(self):  
 key = **""  
 for** i **in** range(self.key\_length):  
 current\_key\_candidates = []  
 **for** c\_table **in** self.alphabet:  
 min\_d = 100.0  
 min\_c = **None  
 for** c\_freq **in** self.frequency\_tables[i]:  
 current\_d = abs(self.frequency\_tables[i][c\_table] - self.stats\_frequency[c\_freq])  
 **if** current\_d < min\_d:  
 min\_d = current\_d  
 min\_c = c\_freq  
 c1 = self.alphabet\_indexes[c\_table]  
 c2 = self.alphabet\_indexes[min\_c]  
 current\_key\_candidates.append(self.alphabet\_values[  
 (self.alphabet\_indexes[c\_table]  
 + self.alphabet\_indexes[min\_c])  
 % len(self.alphabet)])  
 **import** collections  
 counter = collections.Counter(current\_key\_candidates)  
 print(counter)  
 key += counter.most\_common(1)[0][0]  
 **return** key

**statistics.py**

**def** count\_matches(path1, path2): *#подсчет процента соответствия 2 файлов*  
 f1 = open(path1, **'r'**)  
 f2 = open(path2, **'r'**)  
 matches = 0  
 sum = 0  
 **while** 1:  
 line1 = f1.readline()  
 line2 = f2.readline()  
 **if not** line1 **or not** line2:  
 **break  
 for** i **in** range(len(line1)):  
 **if** line1[i] == line2[i] **and** line1[i] **in** english\_alphabet:  
 matches += 1  
 sum += 1  
 **return** matches \* 100 / sum

1. statistics.frequency\_english [↑](#footnote-ref-1)
2. statistics.frequency\_english1 [↑](#footnote-ref-2)
3. encrypter.VigenereEncrypter.encrypt(self, text, key) [↑](#footnote-ref-3)
4. Файл king.txt [↑](#footnote-ref-4)
5. analyzer.VigenereAnalyzer.add\_matches\_info(self, text) [↑](#footnote-ref-5)
6. analyzer.VigenereAnalyzer.count\_matches(self) [↑](#footnote-ref-6)
7. analyzer.VigenereAnalyzer.set\_key\_length(self, key\_length) [↑](#footnote-ref-7)
8. analyzer.VigenereAnalyzer.add\_frequency\_info(self, text) [↑](#footnote-ref-8)
9. analyzer.VigenereAnalyzer.find\_key(self) [↑](#footnote-ref-9)
10. analyzer.VigenereAnalyzer.make\_replacement\_tables(self) [↑](#footnote-ref-10)
11. analyzer.VigenereAnalyzer.decode(self, text) [↑](#footnote-ref-11)
12. statistics.count\_matches(path1, path2) [↑](#footnote-ref-12)